IN THE CLAIMS

What is claimed is:

1- 7 (Deleted).

8 (Currently Amended) A method for managing a memory, comprising:

providing a ~~set~~ plurality of available memory blocks, each memory block having a size;

providing a tree comprised of elements where each element has between 0 and 2 branches inclusive, a value, and an indicator, the indicator enabled to indicate if an element is further a hook-element, each hook-element comprising a first branch to a linked list where each element in the linked list has a value the same as the hook-element value, and a second branch to an element with a same value as the hook-element, and where each of the plurality of available memory blocks is associated with one element with the size of the associated memory block being equal to the value of the associated element;

requesting an allocation of memory comprised of a request size;

searching for one of the available memory ~~block that minimally~~ blocks that satisfies the allocation request, the searching process further comprising:

comparing the request size to a first value of a first element ~~of the allocation request to an element in a binary search tree~~;

determining if the indictor indicates the first element is a hook-element and if the first value is equal to the request size;

traversing, if the indicator indicates a hook-element and the first value and request size are equal, to the hook-element's first branch;

traversing, if the indicator indicates a hook-element and the first value and request size are unequal, to the hook-element's second branch;

traversing, if the indicator did not indicate a hook-element and the request size is larger than first value, to a branch comprised of elements with values all greater than the first value;

traversing, if the indicator did not indicate a hook-element and the request size is smaller than first value, to a branch comprised of elements with values all smaller than the first value;

continuing determining and then traversing until an element that is not a hook-element and has a same value as the request value is reached, being a request element thereby;

removing the request element from the tree; and

making the memory block associated with the removed element available for the memory request.

~~determining that the allocation request is minimally satisfied by the value of the element;~~

~~using a list extending from the element, the list identifying duplicative values equal to the value of the element; and~~

~~decrementing by one the number of duplicative value memory blocks listed at the element; and~~

~~allocating one of the memory blocks at the size equal to the value of the element.~~

9 (Currently Amended) The method according to claim 8, further comprising:

~~providing address data for each memory block;~~

~~retrieving a memory address associated with the element; and~~

~~determining that the memory address is a negative number~~

providing each element with an address; and

indicating an element is a hook-element by providing a negative number as the address.

10 (Deleted)

11 (Currently Amended) The method according to claim 8, wherein the being equal to the value of the associated element further comprises being equal to the absolute value of the associated element, and the indicator indicates the element is a hook-element when the value is negative ~~element is identified as a negative number and the value of the element represents an absolute value~~.

12-14 (Deleted)

15 (Currently Amended) A ~~process~~ method for updating a binary search tree residing in computer memory accessible by a processor responsive to executable instructions, comprising:

providing a binary search tree comprised of nodes where each node has up to 2 branches, a value, and an indicator, the indicator enabled to indicate if a node is further a

hook-node, a hook-node comprising a first branch to a linked list and a second branch to a node with a same value as the hook-node, where each node in the linked list has a same value equal to the value of the hook-node;

~~organizing a set of values in a binary search tree, each unique value being represented by a node element;~~

receiving a new value to be added to the binary search tree ~~set of values~~;

determining that the new value is duplicative of the value of one of the nodes;

identifying a subtree ~~for the node having the duplicative value~~, where the top node of the subtree is the node with the duplicative value;

identifying an attachment point, the attachment point being where a branch is attached to the top of the node with the duplicative value, if there is a branch currently attached to the top of the duplicative node;

~~replacing the node~~ inserting a hook-node above the top node of the substree, ~~having the duplicative value with a hook~~ the hook-node indicating the duplicative value;

extending the subtree from the ~~hook~~ second branch of the hook-node; ~~the subtree having a base parent node at the duplicative value; and~~

extending a ~~list~~ linked list from the ~~hook~~ first branch of the hook-node, the ~~list~~ linked list identifying one of the duplicative values; and

attaching the attachment point to the top of the hook-node, if there was an attachment point.

16 (Currently Amended) The ~~process~~ method according to claim 15, wherein replacing the node with the ~~hook~~ hook-node includes changing the sign of ~~a data point associated with~~ the ~~new~~ value.

17 (Currently Amended) The ~~process~~ method according to claim ~~16 wherein the data point is~~ 15 where the indicator is a memory address.

18 (Currently Amended) The ~~process~~ method according to claim ~~15, wherein replacing the node with the hook includes changing the sign of the duplicative value~~ 17, where the indicator indicates the node is a hook-node if the memory address is negative.

19 (Currently Amended) The ~~process~~ method according to claim 15, further including:

receiving a second new value to be added to the set of values;

determining that the second new value is duplicative of the value of the ~~hook~~ hook-node; and

adding ~~an indication to the list that another duplicative value has been added to the set of values~~ a new node to the linked list branching from the hook-node.

20 (Currently Amended) A handheld portable device, comprising:

RAM memory;

input and output subsystems;

~~an embedded~~ a processor; and

a binary search tree engine ~~implement~~ implementing a memory management process further comprising:

receiving a request for an allocation of memory comprised of a request size;

searching for an available memory block that ~~minimally~~ satisfies the allocation request, the searching process further comprising:

traversing a binary tree, the tree comprised of elements where each element has between 0 and 2 branches inclusive, a value, and an indicator, the indicator enabled to indicate if an element is further a hook-element, a hook-element comprising a first branch to a linked list and a second branch to an element with a same value as the hook-element, and where each of a plurality of available memory blocks is associated with one element with a size of the associated memory block being equal to the value of the associated element;

comparing the request size to a first value of a first element;

determining if the indictor indicates the first element is a hook-element and if the first value is equal to the request size;

traversing, if the indicator indicates a hook-element and the first value and request size are equal, to a linked list of elements having a same value as the request size;

traversing, if the indicator indicates a hook-element and the first value and request size are unequal, to an element having a same value as the request size;

traversing, if the indicator did not indicate a hook-element and the request size is larger than first value, to a branch comprised of elements with values all greater than the first value;

traversing, if the indicator did not indicate a hook-element and the request size is smaller than first value, to a branch comprised of elements with values all smaller than the first value;

continuing determining and then traversing until an element that is not a hook-element and has a same value as the request value is reached, being a request element thereby;

removing the request element from the tree; and

making the memory block associated with the removed element available for the memory request.

~~comparing the size of the allocation request to an element in a binary search tree;~~

~~determining that the allocation request is minimally satisfied by the value of the element;~~

~~using a list extending from the element, the list identifying duplicative values equal to the value of the element; and~~

~~decrementing by one the number of duplicative value memory blocks listed at the element; and~~

~~allocating one of the memory blocks at the size equal to the value of the element.~~

21 (Original) The device according to claim 20, further comprising a transceiver and an antenna.

22 (Currently Amended) A method of searching in a binary search tree, comprising:

moving to a new element in a binary search tree, the element having a value;

retrieving a data flag associated with the new element;

determining whether the data flag is set;

identifying, responsive to the determining step, that the new element is a hook element, the hook element ~~being capable of~~ having two branches comprising a first branch to a linked list and a second branch to an element with a same value as the hook element; and

making a value comparison to the value .

23 (Currently Amended) The method according to claim 22, further comprising:

providing a list of duplicate members on ~~one~~ the first branch ~~of the hook~~; and

allocating, responsive to making the comparison, one member from the list of duplicate members.

24 (Currently Amended) The method according to claim 22, further comprising:

providing a list of duplicate members on ~~one~~ the first branch ~~of the hook~~; providing a subtree on the ~~other~~ second branch of the hook; and

moving, responsive to making the comparison, to an element on the subtree.

25 (Original) The method according to claim 22, wherein the data flag is a memory address.

26 (Original) The method according to claim 22, wherein the data flag is the sign bit for a memory address.

27 (Original) The method according to claim 22, wherein the data flag is the sign bit for the value.

28 (Currently Amended) A binary search tree residing in computer memory readable by a processor, and traversable as instructed by instructions being executed by the processor, comprising:

a node element having a ~~node~~ first value and two branches, comprising:

a first branch having only values that are greater than the ~~node~~ first value; ~~and~~

a second branch having only values that are less than the ~~node~~ first value; and

a hook element having a ~~hook~~ second value and two branches, comprising:

a first branch comprised of a linked list and having only values that are equal to the ~~hook~~ second value; ~~and~~

a second branch ~~having values that are not equal to the hook value~~ comprised of an element with a value equal to the second value.

29 (Currently Amended) The binary search tree according to claim 28, wherein the second branch of the hook element includes a subtree, where the root of the subtree is the element comprised of a value equal to the second value.

30 (Deleted)